Allowed claim 29 has been amended to parallel allowed claim 23, but for an intranet, and new claims 66-70 have been added to parallel claims 24-28.

The remaining claims were rejected under 35 U.S.C. § 102 as being anticipated by Cole et al. (5,752,042) or under 35 U.S.C. § 103 as being unpatentable over Cole et al. in view of He (5,734,898), Ha (5,721,911) and/or Moore (5,732,266). It is respectfully submitted that the rejection cannot be properly applied to the amended claims.

The invention is directed to a method and system for updating application program components on a client through a network. In the preferred embodiment, the client includes a program which performs all update processing, the server system only being required to download files. Specifically, a client which is to update the application program components first downloads a catalog file from the server. The catalog file identifies the most recent versions of the program components as well as the network addresses where those components are stored. The program in the client then compares the version identifications in the catalog file with versions then on the client and, where a new version is required, downloads the new version from the network address indicated in the catalog file. In the preferred implementation, the updating program is a launcher program which serves as a proxy to update the application program when a user selects the application program for execution.

Since all processing is performed at the client, the server need only make available the catalog files and application program components for standard file transfer. No proprietary software is required on the server, and the required catalog and component files can be stored on any file server accessible to the client. Further, the only communications required between the client and the server are to first download the catalog file and to subsequently download any required components.

Prior methods and systems for updating application program components have involved complex interactions between the server and client systems including processing at the server. In very large networks, where many clients may require updating often, the task of processing all of the updates becomes extremely burdensome on the server. Further, the processing burden which would be imposed on the server can only be shared by other servers having proprietary updating software. With the present invention, the server need only respond to file transfer requests, a very routine matter, and the system manager need only make updates in the component and catalog files on the server. Any processing during the update is initiated by and performed at the client which requires the updating.

The principal reference on which the Examiner relies, Cole et al. 5,752,042, relates to a system for updating an entire client system, not components of programs to be executed. Although Cole et al. similarly relies on HTTP and FTP protocols, it differs in the complexity of the updating approach and in that the Cole et al. approach is a server-driven system; whereas, the present invention is a relatively simple client-driven system. In the update process of Figures 3(a) and 3(b) of Cole et al., the client first provides current level information to the server at 104, and the server then determines the necessary update routines required by the client and forwards the addresses for those routines at 107. The client then downloads the basic routines that are required for updating at 108 and 109 and sends additional information to the server at 110.

Based on the basic system information provided by the client, the server determines at 114 all of the programs which might be updated on the client. For each of those programs, the server then sends to the client the address of individual recognizer programs at 111. At 118, the client then obtains the recognizer programs and, for each program to be updated, determines whether the most recent version is located on the client. The results of all the recognizer programs are forwarded to the server at 120. The server then sends the selection form indicated at column 6, lines 9-22 to the client. From that form, the user selects the minimum number of required code updates, the maximum number of such updates or only specific code updates. Only after that selection is sent to the server at 131 does the server return the FTP addresses for the selected updates at 132.

From Figures 3(a) and 3(b) of the Cole et al. patent, it can be seen that the update process is very server intensive. First, the server identifies the updating programs to be downloaded to the client. Then, the server identifies all programs that the client might want to have updated. For each of those programs, the server provides the client with an address to a recognizer program, and the client downloads and processes the recognizer program. Once the client has checked version identifications (in the form of file sizes or version numbers), the server determines the criticality of all of the programs which require updated versions. Only then does the user select which programs to update. The server then supplies the necessary file addresses.

By contrast, the update process of the present invention may be invoked for only the program of interest to the user, and that program is identified at the client, not at the server. Further, the complete updating program resides on the client, so only a simple file transfer of the catalog is required by the server to make the version comparison. Based on information in that

catalog, the client is able to determine whether it has the most recent version and, if not, obtain the most recent version through another simple file transfer.

Nowhere does Cole et al. transfer a catalog file of program components to the client. For that feature, the Examiner references column 3, lines 14-27 of Cole et al. However, that portion of Cole et al. deals with elements 104, 106 and 107 of Figure 3(a). The server merely determines the basic update routines required by the client and provides no information specific to the program. The function served by the catalog file of the present invention is more closely met by the recognizer programs of elements 111 through 120, but through a much more complicated mechanism. For each client program that the Cole et al. server determines might require update, a recognizer program is identified. That recognizer program includes both the updating software, which performs a function served by the launcher program already residing on the client with the present invention, and the program specific information such as version. Even once the recognizer program has completed its routine, the updating process is not complete. Rather, the comparison results are forwarded to the server which must create a form based on the criticality of the programs to be updated. The server forwards that form to the client for a final decision from the user before the server forwards FTP addresses.

The He reference was cited for its use of a cache in a client server environment. However, He does not relate to a system for updating application program components and cannot be said to suggest caching of a component catalog in such an updating system.

Ha et al. is cited for creating a catalog file and using a launcher program to execute application programs. First, there is no suggestion in Ha et al. of a catalog file of components of an application program, the catalog file including version identifications of the components. Ha et al. is not a program updating system. Rather, Ha et al. relates to a catalog system for a database. There would be no reason to combine Ha et al. with the program updating system of Cole et al., and even if the references were combined, there is still no teaching of the claimed catalog.

Ha et al. and Moore et al. are both cited for teachings of launchers, Moore et al. for the step of launching a program from a designated address. Launching programs from designated addresses is certainly well known in the art and is not by itself a feature of the present invention. Rather, the present invention relates to updating such a program through a network during the launch process. Neither Ha et al. nor Moore et al. relates to such a program updating process.

Amended claim 1 recites that a catalog of application programs components is maintained on the server, a feature not suggested by the references cited. Further, in accordance with claim 1, the client identifies an application program for update and downloads the associated catalog. In Cole et al., the only cited prior art relating to program update, the server identifies a list of programs to be considered for potential updating, thus requiring special update processing at the server. As such, Cole et al. fails to teach the client driven update process recited in claim 1. Further, claim 1 explicitly recites that substantially all processing for updating the application program is performed at the client. As discussed above, a substantial portion of the update process of Cole et al. is performed at the server, thus requiring proprietary software and substantial processing time at the server and extensive communications between the server and client during processing.

Claim 31 also recites the client driven update system. Specifically, a user selects an application program to execute the program, and the client causes the server to download a component catalog for that application program. Accordingly, only the limited update processing required by a client is performed, unlike the extensive processing of Cole et al.

Independent claim 60 recites the client which includes the software required for the update in a client driven update. Specifically, claim 60 recites a launcher program which responds to a user selecting an application program. The launcher program on the client causes the server to download a component catalog, makes the version comparison and updates the application program by downloading select components. Such a client is not suggested by Cole et al.

Claim 61 recites the program for execution on the client in a client driven update. Again, that program operating on the client causes a component catalog to be downloaded when a user selects an application to be executed, compares version identifications and downloads an appropriate update. Such a program for execution on a client is not suggested by Cole et al.

Claim 62 recites in detail the preferred method of the present invention. The server stores the application components and the file catalog for retrieval through a standard file transfer request. A launcher program on the client operates as a proxy for each of the applications program to retrieve a component catalog file for an application program, retrieving the required components, installing the retrieved components and storing the retrieved catalog file for use in a subsequent update process. As discussed above, there is no component catalog in Cole et al. Further, there is no program stored on the client to operate as a proxy for each application

program. Still further, there is no storage of a catalog file from one update process for use in a subsequent update.

Claim 65 also recites in detail a preferred method of the invention. Specifically, claim 65 recites that the downloaded component catalog is retained on the client for comparison to a second catalog downloaded in a subsequent updating process. Cole et al. does not suggest downloading of the component catalog, much less storing that catalog for comparison to a subsequently downloaded catalog.

CONCLUSION

In view of the above amendments and remarks, it is believed that all claims are in condition for allowance, and it is respectfully requested that the application be passed to issue. If the Examiner feels that a telephone conference would expedite prosecution of this case, the Examiner is invited to call the undersigned at (781) 861-6240.

Respectfully submitted,

HAMILTON, BROOK, SMITH & REYNOLDS, P.C.

By _____

James M. Smith, Esq.
Registration No. 28,043
Telephone (781) 861-6240
Facsimile (781) 861-9540

Lexington, Massachusetts 02421-4799
Dated: 11/13/98